Session 135

Input for Games

Geoff Stahl
Graphics & Games DTS Engineer

What You'll Learn

- System services for input
  - Carbon
    - Basic Overview
  - HID Manager
- Getting user input
  - Device discovery
  - Retrieving input
  - Input configuration

System Services for Input

- Mac OS X
  - HID Manager
    - On top of IOKit, handles device communication
    - Joystick, Jog Shuttle, UPS, Gamepad
  - Carbon
    - Handles mouse and keyboard
- Future
  - Mouse and keyboard through HID Manager

Carbon

- Carbon Events
  - Allows access to multiple buttons and mouse deltas
- Carbon
  - **GetKeys ()** and **GetMouse ()** still supported

Carbon Events

- Mouse
  - **kEventMouseDown, kEventMouseUp, kEventMouseWheelMoved, kEventMouseDragged, kEventMouseMoved**
    - **kEventParamMouseWheelDelta, kEventParamMouseDelta**
  - Disassociating Pointer

    CGAssociateMouseAndMouseCursorPosition (false);
- Keyboard
  - **kEventRawKeyDown, kEventRawKeyRepeat, kEventRawKeyUp, kEventRawKeyModifiersChanged**

HID Manager

- Device discovery via IO Kit services
- Allows access to device elements
  - Axis, buttons, directional pads, sliders, dials
- Provides polling and queuing for device elements
  - Element centric design

Sidebar: API Comparison

- HID Manager
- DirectInput
- InputSprocket

API Comparison

	Device Discovery
HID Manager	Use IOKit to iterate devices and then elements; Save element cookies
DirectInput	Create DirectInput object. Enumerate Devices; Create DirectInputDevice object
InputSprocket	Use ISpDevices_Extract or not needed with ISpInit

API Comparison

	Element Handling
HID Manager	Use HID device interface and cookie found through IOKit to access element
DirectInput	Use EnumObjects and/or GetCapabilities to identify elements
InputSprocket	ISpDevice_GetElementList to retrieve list of elements; Obtain ISpElementReference from list

API Comparison

	User Configuration
HID Manager	Handled by application; Save ProductID, VendorID and Location for positive device ID
DirectInput	Handled by application; Save combination of device info for device ID
InputSprocket	Handled via ISpConfigure for high level interface otherwise handled by application

API Comparison

	Input Retrieval
HID Manager	Use getElementValue for pollng or getNextEvent for event handling on a per element basis
DirectInput	Use GetDeviceState or GetDeviceData to retrieve data for entire device; Application must parse as required
InputSprocket	Retrieve value using ISpElement_GetSimpleState or ISpElement_GetNextEvent

Getting User Input

- Device Discovery
  - Set up I/O matching dictionary
  - Get device iterator
  - Iterate for devices
  - Recursively iterate under device element key
    - CFArray
    - CFDictionary
  - Capture element cookies for element identification

Device Discovery

- Set up I/O matching dictionary

```
CFNumberRef refUsage = NULL, refUsagePage = NULL;
CFMutableDictionaryRef hidMatchDictionary = NULL;

// match devices by class name for all HID class devices.
hidMatchDictionary = IOServiceMatching (kIOHIDDeviceKey);

// Add key for device type to refine the matching dictionary.
refUsage = CFNumberCreate (kCFAllocatorDefault, kCFNumberIntType,
    &usagePage);
refUsagePage = CFNumberCreate (kCFAllocatorDefault,
    kCFNumberIntType, &usage);
CFDictionarySetValue (hidMatchDictionary , CFSTR
    (kIOHIDPrimaryUsagePageKey), refUsagePage);
CFDictionarySetValue (hidMatchDictionary , CFSTR
    (kIOHIDPrimaryUsageKey), refUsage);
```

Device Discovery

- Get device iterator

```
io_iterator_t hidObjectIterator = NULL;
mach_port_t masterPort = NULL;
// get master I/O port
IOMasterPort (bootstrap_port, &masterPort);
// Set up matching dictionary
hidMatchDictionary = SetUpHIDMatchingDictionary (usagePage, usage);
// search I/O Registry for matching devices.
IOServiceGetMatchingServices (masterPort, hidMatchDictionary,
    &hidObjectIterator);
// IOServiceGetMatchingServices consumes a reference to the dictionary
hidMatchDictionary = NULL;
```

Device Discovery

- Iterate for devices and get properties

```
io_object_t ioHIDDevice = NULL;

while ((ioHIDDevice= IOIteratorNext (hidObjectIterator)))
{  // get dictionary for HID properties
    CFMutableDictionaryRef hidProperties = 0;
    IORegistryEntryCreateCFProperties (ioHIDDevice, &hidProperties,
        kCFAllocatorDefault, kNilOptions);
    MyGetDeviceInfo (ioHIDDevice, hidProperties);
    MyGetCollectionElements (hidProperties);
    CFRelease (hidProperties);
    // don't release device iterator object since we will need it later.
}
IOObjectRelease (hidObjectIterator);
```

Device Discovery

- Recursively iterate under device element key

```
void MyGetCollectionElements (CFMutableDictionaryRef properties)
{
        CFTypeRef refElement = CFDictionaryGetValue (properties, CFSTR
            (kIOHIDElementKey));
        CFTypeID type = CFGetTypeID (refElement);
        if (type == CFArrayGetTypeID ())
        { // if element is an array
            CFRange range = {0, CFArrayGetCount (refElement)};
            // MyGetArrayElements called for each array member
            CFArrayApplyFunction (refElement, range, MyGetArrayElements,
                0);
        }
}
```

Device Discovery

- Recursively iterate under device element key (2)

```
void MyGetArrayElements (const void * value, void * parameter)
{
    if (CFGetTypeID (value) == CFDictionaryGetTypeID ())
    { // if dictionary type
        CFTypeRef refType = CFDictionaryGetValue ((CFTypeRef) value,
          CFSTR (kIOHIDElementTypeKey));
        if ((elementType == kIOHIDElementTypeInput_Misc) ||
          (elementType == kIOHIDElementTypeInput_Button) ||
          (elementType == kIOHIDElementTypeInput_Axis))
            MyGetElementInfo (value); // get info for particular element
        else if (elementType == kIOHIDElementTypeCollection)
            MyGetCollectionElements ((CFMutableDictionaryRef) value);
    }
}
```

Device Discovery

- Capture element cookies

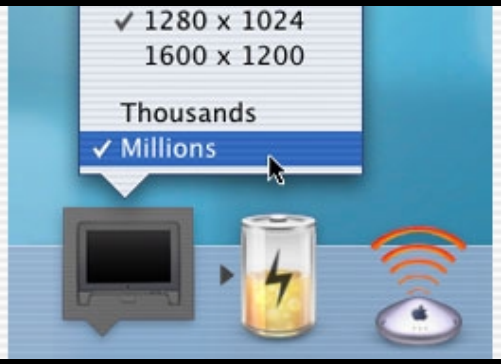
```
void MyGetElementInfo (CFTypeRef refElement) {
    IOHIDElementCookie cookie;
    CFTypeRef refType;
    refType = CFDictionaryGetValue (refElement, CFSTR
    (kIOHIDElementCookieKey));
    if (refType && CFNumberGetValue (refType,
    kCFNumberLongType, &((long) cookie) ))
}
```

- Get other information of interest (IOHIDKeys.h)

  type, usage page, usage, min, max, scaled min, scaled max, size, relative, wrapping, non-linear, preferred state, null state, units, unit exp, name

Demo

Device Discovery



Getting User Input

- Retrieving Input
  - Setting up the interface
  - Polling
  - Event Queues
  - Carbon Events

Retrieving Input Example

- Setting up the device interface

```
SInt32 score = 0;
IOCFPlugInInterface ** ppPlugInInterface = NULL;
IOHIDDeviceInterface ** pphidDeviceInterface = NULL;
IOCreatePlugInInterfaceForService (ioHIDDevice,
    kIOHIDDeviceUserClientTypeID, kIOCFPlugInInterfaceID,
    &ppPlugInInterface, &score);
// use intermediate plug-in to create the device interface
plugInResult = (*ppPlugInInterface)->QueryInterface (ppPlugInInterface,
    CFUUIDGetUUIDBytes (kIOHIDDeviceInterfaceID), (void *)
    &pphidDeviceInterface);
(*ppPlugInInterface)->Release (ppPlugInInterface);
// open interface
(*pphidDeviceInterface)->open (pphidDeviceInterface, 0);
```

Retrieving Input Example

- Polling

```
IOHIDEventStruct hidEvent;
hidEvent.value = 0;

(*(pphidDeviceInterface))->getElementValue (pphidDeviceInterface ,
    cookie, &hidEvent);

    return hidEvent.value;
```

Retrieving Input Example

- Event Queues: Build queue and add element

```
IOHIDQueueInterface ** queue;

// build queue
queue = (*(pphidDeviceInterface))->allocQueue (pphidDeviceInterface);
if (queue)
    (*queue)->create (queue, 0, kDeviceQueueSize);

// queue element
(*queue)->addElement (queue, cookie, 0);

// start queue
(*queue)-> start (queue);
```

Retrieving Input Example

- Event Queues: Retrieve events

```
Boolean MyGetEvent (IOHIDQueueInterface ** queue, IOHIDEventStruct *
    pHIDEvent)
{
    IOReturn result = kIOReturnSuccess;
    AbsoluteTime zeroTime = {0,0};

    result = (*queue)->getNextEvent (queue, pHIDEvent, zeroTime, 0);
    return true;
}
```

Retrieving Input Example

- Carbon Events (mouse location)

```
if ((GetEventKind (event) == kEventMouseDragged) ||
        GetEventKind (event) == kEventMouseMoved))
{
    Point movePoint;
    GetEventParameter (event, kEventParamMouseLocation,
            typeQDPoint, NULL, sizeof(Point), NULL, & movePoint);
        UpdateInput (movePoint);
}
{
    Point deltaPoint;
    GetEventParameter (event, kEventParamMouseDelta,
            typeQDPoint, NULL, sizeof(Point), NULL, & deltaPoint);
    UpdateInput (deltaPoint);
}
```

Retrieving Input Example

- Carbon Events (mouse button)

```
if (GetEventKind (event) == kEventMouseDown)
{
        Point downPoint;
        UInt32 modifiers;
        EventMouseButton button;
        GetEventParameter (event, kEventParamMouseLocation,
                typeQDPoint, NULL, sizeof (Point), NULL, & downPoint);
        GetEventParameter (event, kEventParamKeyModifiers,
                typeUInt32, NULL, sizeof (UInt32), NULL, &modifiers);
        GetEventParameter (event, kEventParamMouseButton,
                typeMouseButton, NULL, sizeof (EventMouseButton),
                NULL, &button);
}
```

Retrieving Input Example

- Carbon Events (keyboard)

```
if ((GetEventKind (event) == kEventRawKeyDown) ||
        (GetEventKind (event) == kEventRawKeyRepeat))
{
        char keyChar;
        UInt32 modifiers;
        GetEventParameter (event, kEventParamKeyMacCharCodes,
                typeChar, NULL, sizeof (char), NULL, & keyChar);

        GetEventParameter (event, kEventParamKeyModifiers,
                typeUInt32, NULL, sizeof (UInt32), NULL, &modifiers);
}
```

Demo

Retrieving Input

Getting User Input

- Input configuration
  - Build application input “actions” list
  - Present UI to associate action with element
  - When user selects action to assign
    - Build queues for devices of interest
    - Retrieve first element event/poll
    - Save element cookie to associate with action
  - Use cookie list for input
  - Saving configurations

Saving Configurations

- Element
  - Cookie
  - Usage page, usage
- Device
  - Serial number
  - Vendor ID, product ID, location
  - Usage page, usage

Matching Configurations

- Vendor ID, product ID, serial number, and cookie
- Vendor ID, product ID, device location, and cookie
- Vendor ID, product ID, and cookie
- Usage page and usage
  - Search for element by usage page and usage
- Any device
  - Search for element by usage page and usage

Demo

Input Configuration

Roadmap

136 Sound & Networking for Games	Room J2 **Fri., 2:00pm**
FF120 Feedback Forum: Games Technologies	Room J1 **Fri., 5:00pm**

Who to Contact

Geoff Stahl
Graphics & Games DTS Engineer
Apple Worldwide Developer Relations
gstahl@apple.com

Q&A

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001